Trusted Computing Based Collaborative Intrusion Detection System

· Hadi KhorasaniZadeh [a, b], Norbik Bashah Idris [a], Jamalul-Lail Ab Manan[b]

[a]*Advanced Informatics School, University Technology Malaysia, Kuala Lumpur, Malaysia. khadi4@live.utm.my, norbik@ic.utm.my*
[b]*Advanced Analysis & Modeling, MIMOS Berhad, Technology Park Malaysia, Kuala Lumpur, Malaysia, Jamalul.lail@mimos.my*

Abstract

Collaboration and information sharing has obliged participating parties to look for improved detection accuracy and reaction speed in Distributed Intrusion Detection Systems (DIDS) solutions. This is mainly due to the increasing number of attacks as well as increasingly sophisticated intrusions and more alarmingly various critical components of a system can be targeted. This is further exasperated by the fact that most DIDS models do not consider the attacks targeting the collaborative network itself. We specifically find this issue to be very critical and hence in this paper we propose a trust aware DIDS simulation model that is capable of categorizing each participating IDS expertise (i.e. speciality and competence), therefore helps collaborating organizations to consult our simulation model for choosing the right candidate for any type of intrusion. We call our Model as Consultative Trusted Computing-based Collaborative IDS (CTC IDS). We utilize the Trusted Platform Module (TPM) for integrity evaluation and fine-tuning peer evaluation.

Keywords: Distributed Intrusion Detection Systems, Collaborative Intrusion Detection System, Trust Management in Distributed Intrusion Detection

1. Introduction

Since the early days of computer networks intrusion detection systems have been around and evolved dramatically since their early introduction (by Anderson 1980[1]). Intrusion detection is by definition, the art of detecting network intruders by utilizing any source of information. In general, an Intrusion Detection Systems (IDS) can take various forms such as hardware or software, intelligent based or statistical based, signature or anomaly based, networked or host based, real-time or off-line, active response or passive response and finally centralized or distributed. Over the many years after its inception, IDS evolution took different approaches and viewpoints, each trying to enrich the process from their own point of view.

In the past decade, because of the enhancements in networking and the vast growth of internet the study on distributed IDS has been the main area of research. New types of attacks appeared and worms that infected thousands of networks within hours and became part of everyday challenges. One of such worms was Code Red that infected 250,000 computers in just 8 hours[2]. It has been proven that distributed intrusion detection systems facilitate better detection accuracy and increased speed in contrast to the conventional detection systems [3-7].

Ideally, a Distributed Intrusion Detection System (DIDS) should be able to cross the boundaries of operating systems and vendors. Ideally, a DIDS should be able to talk to one another peer DIDS and exchange information with complete trust and confidence. Through this trusted collaboration amongst DIDS, they should be able to extract and discover any new attacks faster and with greater precision.

As discussed above, the most crucial part of DIDS is the trust amongst participating members. In most proposed models this fact has been neglected and collaborating members

have been assumed to trust each other in sharing legitimate information. We view that, Collaborative Intrusion Detection Systems (CIDS) model is highly desirable but without control over received data from anonymous user, it is prone to serious attacks targeted at the intrusion detection network itself. A malicious peer, for example, could create biasness over others towards unstable security state, such as black listing legitimate hosts or perceiving legitimate traffic as illegitimate, hence causing a disastrous denial of service (DOS).

Our main focus on this paper is on how to minimize the risk of information sharing amongst IDS in the network, without affecting their main duty of detecting intrusions. We propose utilization of the Trusted Platform Module (TPM) to improve by imposing trust policy on individual nodes. We are aware that not every entity is willing to or has the TPM hardware enabled (TC-enabled); therefore we assumed a heterogeneous network made of systems with and without this hardware functionality. In this paper, we evaluate the IDS heterogeneous network (TC-enabled or otherwise), in order to estimate the amount of trustworthiness of each node, as well as their expertise field of operation.

We organize the rest of this paper as follows: (1) a review of current DIDS work, (2) an overview of trusted computing platform, (5) types of threats in distributed computing, (4) our proposed model (5) evaluation and analysis and (6) conclusion and future work.

2. Distributed Intrusion Detection

In this section the different approaches to Distributed intrusion detection are studied. First we will study their architecture, followed by the challenges and attacks against these systems.

2.1. Architecture

In this section our focus is on the information collaboration architecture and the ways that participating hosts exchange information. There are three types of distributed intrusion detection systems namely, Centralized systems, Hierarchical systems and Fully Distributed systems. Each of these models has strength in a particular field and implementation.

2.1.1. Centralized systems

Older versions of distributed intrusion detection systems that relied on a central point of analysis include ASAX[8], DIDS (Abraham, Jain et al. 2007) and NSTAT(Abraham, Jain et al. 2007) [9]. All these systems gather data from different sources and analyze it in a different manner. They all share a similar general structure in which information is sent to a central point for analysis and feedback. This structure can be hazardous as this central point of analysis is prone to failure, attacks and misuse. The result would compromise the entire distributed architecture. Furthermore, such systems are prone to limited scalability as the central point can only handle limited number of nodes.

The main advantage of having a central control point structure is faster detection and reaction time, as all peers are interacting directly with it. Similarly, the central command point is aware of all the events happening on every participating host. Such systems can be used where a limited number of hosts collaborate and a single server can handle all requests. However, the single point of failure drawback remains its main disadvantage, which is usually handled by redundancy.

2.1.2. Hierarchical systems

To improve the scalability problem in DIDS, a number of new systems were proposed using hierarchical structures, such as the following implementations; EMERALD (Phillip A. Porras 1997)[10], GriDS (Steven Cheung, Jeremy Frank et al. 1999) [11] and AAFID(Mark Crosbie 1995) [12]. For these proposals, each of them creates a hierarchy using a proposed algorithm. A tree is formed and messages are sent from bottom to top of the tree and feedback is provided from top to bottom. It is obvious that, the problem of having a central

point of failure still remains but the system could handle expansion to any size.

Another disadvantage of the hierarchical architecture is that it ultimately relies on a central component; therefore, it is not truly distributed. [5] The other setback with this scheme is; if two IDSs which are far apart in the hierarchy, concurrently detect an intrusion, the results cannot be correlated until it reaches a certain higher level of IDSs. Moreover, there is a delay imposed on the system, as each level in the hierarchy may have its own latency to update, alerts any correlation with a higher level, which might result in a much longer time to reach a decision point as compared to centralized architecture.

The EMERALD [10] project uses both signature and statistical anomaly intrusion detection. The GriDS gathers computer and network data into activity graphs that will demonstrate the causal structure of network activity. The Common Intrusion Detection Framework (CIDF) [13] proposes a model that can communicate with different intrusion detection and response components and is capable of sharing information and resources in a distributed environment. The Coordinated and Response System (CARDS) proposes a system that can detect distributed attacks that cannot be normally detected using data from a single host [9]. The Autonomous Agents for Intrusion Detection (AAFID) [12] utilizes autonomous agents for data collection and analysis. It places agents hosted on network nodes, filters for pertinent data extraction, transceivers for supervising agent operations and monitors to collect reports from transceivers.

From the above discussion, we view that creating a hierarchy can bring new improvements in the DIDS model, in comparison to the centralized version. However, it still needs redundancy, anonymity, and robustness against system failure.

2.1.3. Fully Distributed systems

Recent DIDS systems proposals such as LarSID [4], Locasto et al. [7] and VIGILANTE rely on distributed architecture to exchange alert information.

The LarSID (Large scale Intrusion Detection) is based on peer to peer approach and uses Distributed Hash Table (DHT) to distribute possible intrusion information amongst peers. This model has been simulated and tested and shows improved performance over singular detection method. However, there is no weightage system proposed, i.e. each and every incident is viewed with same weightage, and the importance of local or regional data has not been considered. Furthermore, the proposed system did not take into consideration the trust between anonymous peers. [3]

Locasto et al [7] also proposed a Peer to Peer (P2P) approach that uses Bloom filters to preserve privacy among peers. A dynamic overlay network is used for distributed correlation. Each participating IDS generates a watch-list and shares the compressed list with other peers. This model focuses on efficient information exchange using a distributed correlation scheduling algorithm. This strategy improves the bandwidth usage but may cause false positive due to the Bloom filters usage [4].

Vigilante is a collaborative worm containment system. Each participating host in the system runs worm detection software and broadcasts alerts all host via the Pastry overlay. The filters on individual host are then updated based on the received alerts.

The latest works that are similar in concept to our paper is from Fung et al. [14-17] who tried to address the concerns relating to trust management. In their proposal, each node in the framework evaluates its peers based on consultation result which it received. The system then generates a list of acquaintances to which it forms different trusts levels, which is based on a threshold. From the list of acquaintances, the most trusted nodes responses will have further influence on a specific alert ranking. Also discussed in this model are the incentives for nodes to participate and support others.

2.2. Challenges In DIDS

The use of distributed intrusion detection has the following drawbacks that need to be addressed as follows:

- Network traffic: A large amount of network traffic is generated and shared amongst hosts. Hence, usually messages are shortened, hashed, compressed and a hierarchy is used to minimize the amount of network traffic. However, the area of optimizing

the messages traffic is still highly researched.

- Central point of failure: Many proposed models rely on a central or a level based architecture to distribute the load and optimize network traffic. This architecture, as explained earlier, has the risk of being compromised and in most cases they are the ones being attacked. A robust approach is needed to minimize the risks of central point of failure.
- Delays: Alert correlation can have intervals and experience delays in propagating a message to higher levels. These delays can be critical especially during worm outbreaks and distributed attacks. On the other hand, this issue has to be handled carefully not to overwhelm the system, as attackers can create artificial delays by causing the system to broadcast many messages and cause the system to get unstable by its own generated traffic.
- Exposing private information: There is a need for all distributed systems to prevent the private information in IDS from being exposed or leaked to other non participating collaborative IDSs. This has to be carefully designed in any system to make sure no such information is leaked out to non participating IDSs.
- Validation of information: There is a need in all systems to ensure the correctness of the information received from other participating IDSs. This problem has so far been neglected in many proposed solutions and all participants are assumed trusted enough to share legitimate information. We need a way to ensure that each node verifies its integrity of software and the shared data to others.
- Rule updates and policy changes: IDSs need to accept and verify integrity of updates of software, rules and policies when they are sent to it.

2.3. Attacks against DIDS network

It has been discussed before that most studied models in section B, lack trust evaluation and management. We note that in security and trust concerns, it is crucial to ensure validity and trustworthiness of the received information. In a single DIDS concept, issues of lack of trust evaluation and trust management may lead to attacks against the IDS network, or the underlying layer that share this information. If the attacker is able to discover the IDS structure, it could become the target for attacks and make it bias towards any state desired by the attacker. The desired Collaborative IDS (CIDS) itself has to be robust and reliable as it is supposed to operate when every other part fails.

The common threats in distributed systems that have impacted intrusion detection have been studied. An example that would correspond to the type of attack in human society is provided for each as presented below.

- Sybil Attack, A single peer generates fake identities and joins the network as multiple peers. This is comparable to a person pretending to have multiple personalities and interact with many people, when in fact it is only one person.
- Identity Cloning Attack, A peer pretends to be some other peer. This is comparable to a person pretending to be someone else and providing guidance that would benefit his real identity or fulfill his goal.
- Reset to default, An attacker (who has been identified as malicious by the network) can leave and then join the network again with a new identity that is more trusted than before. This is analogous to a thief known to the public, pretending to leave the suburb and come back as a new person not known to anyone.
- Betrayal, An attacker that does not show who he really is joins a network and then after gaining trust and reputation he turns malicious. This is analogous to a trusted person that has behaved trustworthily and changes to be wicked. This can be intentionally (i.e. planned to launch an attack) or unintentionally (i.e. being infected).
- Targeted Betrayal, This refers to a trusted peer which acts maliciously only if certain condition is involved. It is comparable to a person that is untruthful in one condition, but he is trustworthy in every other aspect. Another example can be an attacker trying to conceal his identity and spread his attack to others, but reporting truthfully every other attacker.
- Rumour Attack, This refers to a group of attackers working together report an incident

with a bias (either exaggerate or depreciate) to impose the network into believing that is true.
- Distraction Attack, An attacker over exaggerates an event to raise the detection threshold or flood the network in order to conceal some other less significant attack.
- On-off attack, The attacker performs betrayal or targeted betrayal on and off in a random interval.

3. Trusted Computing

3.1. Introduction

Trusted computing is the art and science of enforcing security requirements and policies on every step of software and hardware design, development, testing and maintenance. Trusted computing group attempts to address software vulnerabilities through hardware administered policy enforcement and control. This group has proposed utilizing Trusted Platform Module (TPM), a tamper-proof hardware with standardized capabilities that enforce security requirements. The TPM chip is commonly deployed on most modern computing systems.

The features of this secure hardware are:
- Integrity measurement, which enables self-integrity measurement as well as report generation of the host operating system and intrusion detection system
- Memory curtaining, which ensures programs can access the allocated memory and do not violate the access privileges.
- Sealed storage, which encrypts and then stores a value on the hard drive, that can be decrypted only when certain conditions are met (integrity of system, correct application running and other conditions)
- Remote attestation, which ensures that the host sending out information to the peers (such as an event report) has not been compromised and all processes pertaining to intrusion detection are running as expected.
- Identity Management, whereby through the use keys generated by the endorsement key, the host can authenticate to the others on the network.

In this paper we will be utilizing the identity management mechanism and remote attestation. The other mentioned features are not directly utilized as they would not affect the network, but rather the host performance.

3.2. Attestation

Attestation is one of the most practical features in trusted platform module. It can assist in distributed computing through the provided services of TPM. Attestation involves a challenger requesting information from the attester that would provide evidence that software and hardware state comply with the challengers' requirements. We may categorize the types of attestation into *code control*, *code analysis* and *delegation* [18]. *Code control* attempts to monitor the well being of another software through its behavior. System calls, I/O and resource usage can be the attested metric in this type of attestation. *Code analysis* checks code integrity either as it is being loaded into memory or during execution. One of the first to implement this type of attestation were Sailer et al. [19] at IBM's T.J Watson research center. They implemented a form of Integrity Measurement Algorithm (IMA). The idea is to evaluate program hashes as they are being loaded into memory for execution. These hashes are stored in a list called Stored Measurement Log (SML) as well as being stored inside TPM's PCRs. The challenger would look up these hashes against known good and bad hash databases. *Delegation* involves a trusted third party attesting certain properties and reporting the result back to the challenger.

4. Our Proposed CTC IDS Model

The distributed intrusion detection network can be built from individual IDS systems (or Host based IDS) as well as Network based IDS that monitor the allocated network/host. They could be based on open source software or any proprietary software/hardware with the help of an adapting software layer to facilitate their collaboration and realize their connectivity to the collaborative network. The collaborating network structure is based on peer to peer communications without any central authority taking control. Individual IDS can join or leave the network at any time.

Our proposed Consultative TC-based Collaborative IDS (CTC IDS) Model will implement means to encourage individuals to remain and gain reputation on the network. We propose the use of TPM in order for peers to ensure the security and validity of other peers, by the functions provided though the standards of trusted computing group. We would also be utilizing Property Based Attestation (PBA) to evaluate different metrics of the remote peers. We assumed, in the near future, all computing systems would be equipped with TPM, or similar hardware based security peripheral. Nevertheless, in every network there are legacy systems involved which may not have the TPM. Furthermore, the utilization of TPM functionalities in most software is left to end users that may opt-in. For the open source based systems and distributed systems, they require flexible and compatible system design; therefore in our proposed work, utilizing the TPM chip would be encouraged but not mandatory.

4.1. Types of Information

In order to realize a successful detection in collaborative IDS, each node would send information to other participating peers. The shared information should keep the individual user and network data private. The peers should be capable of joining and leaving the network at any time. The information types that intrusion detection systems distribute can include:

- *Incident specific data*- Information which is specific to a host/network experience such as attack statistics, IP address of attackers, IP address of under attack hosts, geographical location of attacker/Victim
- *General knowledge*- Information extracted from the incident specific data that could be applied to mitigate similar trends/activity such as ports/applications under attack, signature of new intrusion, the signature of active attacks
- *Consultation* - The suspicious pattern is extracted and sent to the peers for threat evaluation by them and the result is reported back.

4.2. Network control

In our CTS IDS Model, in order to join the network, every peer must first obtain a proof of identity. The peer has to either utilize the TPM and generate identifying keys or register with a Certifying Authority (CA) and receive the signed certificate his public key. Peers can join and leave the network at any time. Any node intending to join has to provide its expected expertise levels to the network before it can join. Once joining process has been successful, it would be provided with a list of other peers and their respective specialization fields.

Just like investing in hardware and software, every peer can invest in their bandwidth and running process that makes them more secure by being trusted and contacted in the collaborative network. (This is similar in social behavior, whereby if a peer can build a reputation and help others, he/she would be helped by others later on)

4.3. Trust Management Model

We build our CTC IDS model by formulizing the following components:

4.3.1. Specializations

It is a known fact that every IDS is capable of detecting certain types of attacks better than others. For example, specific IDS may be capable of detecting windows based vulnerabilities and another could be built specifically to detect worms. This follows the social pattern that everyone trusts an expert in a certain field of knowledge. For example, architects are trusted and consulted for property development. Every participating peer in our proposed CIDS model is expected to be specialized at a particular field. In our model we define *specialization* of every peer as:

$$Sp_{i,j} = \{sp_{i,j,1}, sp_{i,j,2}, \dots sp_{i,j,n}\} \quad (1)$$

$$sp_{i,j,k} \in [0,1]$$

Where $Sp_{i,j}$ is the anticipated *specialization* of peer *j* observed by *i*; *n* is the number of specializations that are being considered by the network; $sp_{i\,j,k}$ is the expertise level of peer *j* observed by *i* in the field of *k*, where $sp_{i,j,k}$=1 denotes highest level of knowledge and $sp_{i,j,k} = 0$ denotes no expertise at all. Every peer advertises its expertise level while joining the network.

4.3.2. Attestation

Every CIDS peer is equipped with hardware based attestation capable of verifying its properties to others. The *properties* being attested are defined as:

$$P = \{p_1, p_2 \dots p_m\} \quad (2)$$

$$A_{i,j} = \frac{\sum_{k:p_k \in P} a_{j,p}.I_p}{\sum_{k:p_k \in P} I_p} \quad (3)$$

$$p_k, a_{j,p}, I_p \in [0,1]$$

Where $p_k$is the *property* being attested, $a_{j,p}$shows the attested property value for node j and $I_p$is a weight representing the importance of this property; *m* is the total number of properties attested and $A_{i,j}$ is attestation outcome of properties of peer j being observed by I which is a value between 0 to 1.

4.3.3. Threat Evaluation

Every node evaluates its peers by asking for information already known beforehand. The result of this evaluation is used to calculate the trust value of its peers. As mentioned earlier, every IDS has an expertise level in every speciality being examined. The peers response to information requested would be evaluated based on its knowledge of the *speciality*, and the competency of the requested data.

$$S_{i,j,t} = \left(1 - \left(Th_{j,t} - Th_{i,t}\right)^2\right) \quad (4)$$

$$S_{i,j,t} = S_{i,j,t}.tanh\left(\rho_1 D_{i,t} - \rho_2.abs\left(Th_{j,t} - Th_{i,t}\right)\right) \quad (5)$$

$$S_{i,j,t} = S_{i,j,t}^{(\sigma_1 + \sigma_2.Th_{i,t})} \quad (6)$$

$$\left(Th_{j,t} + threshold\right) \leq Th_{i,t} \quad (7)$$

In equation (4) and (5), $S_{i\,j,t}$ is the *satisfaction* from a threat evaluation message sent to node j and observed by i. We define *satisfaction* as a general evaluation of received information and do not take the node specific properties such as expertise level and attestation into consideration. $D_{i,t}$ shows the *level of difficulty* of this request. The difficulty can be calculated over the response ratio of the trusted parties (Variance of received messages), where 0 means a very obvious fact and 1 denotes extreme difficulty. $Th_{i,t}$ is the anticipated threat level of test message. The threat level shows the severity of event. $Th_{j,t}$ shows the received threat level in response to the test message. $\rho_1$and ρ_2 are coefficients expressing the peer's sensitivity to inaccurate outcome. Equation (4) results in the value of 1 where $Th_{j\,t} = Th_{i,t}$ and this drops as the two disperse as mentioned in our earlier approach[20]. The formula in (4) never reaches satisfaction of zero. This is amended and the influence of difficulty is applied in equation (5). Where the requested information is easy we are less satisfied with inaccurate responses and as they get difficult, we penalize them more. Figure (1) shows the satisfaction to replies with threat level of 0.8 for different difficulties. The values assigned to ρ_1 and ρ_2 are 8 and 7 respectively.

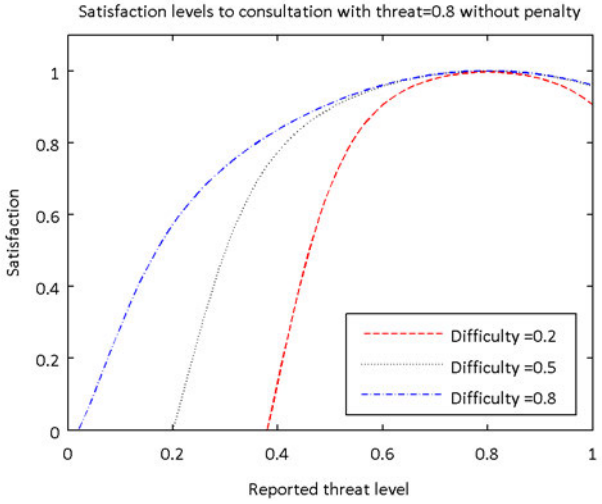

Figure 1 Satisfaction vs. Reported threat for threat level of 0.8

Equation (6) there is a penalty assigned for severe threats being categorizes as unimportant. It can be used where the conditions of (7) are met. σ_1 and $\sigma_2$are coefficients for the amount of penalty imposed.

4.3.4. Trust Calculation

The amount of trust between peers is calculated from a weighted average of satisfaction. The weight is calculated in a way that the replies to the speciality types with more knowledge have the most influence. We also encourage the use of TPM by influencing the final trust value, with the properties attested. The *trust value* for a single threat evaluation is given by:

$$T_{i,j,t} = \frac{\sum_{k:Ty_{i,k}=n}\left[\left(1+Sp_{i,j,Ty_{i,t}}-D_{i,t}\right)^{\left(1-\eta.A_{i,j}\right)}.S_{i,j,k}\right]}{\sum_{k:Ty_{i,k}=n}\left(1+Sp_{i,j,Ty_{i,t}}-D_{i,t}\right)^{\left(1-\eta.A_{i,j}\right)}} \tag{7}$$

$$T_{i,j} = \frac{\sum_{k=1}^{l}\left[\left(1+Sp_{i,j,Ty_{i,t}}-D_{i,t}\right)^{\left(1-\eta.A_{i,j}\right)}.S_{i,j,k}\right].F_k{}^{l-k}.S_{i,j,k}}{\sum_{k=1}^{l}\left[\left(1+Sp_{i,j,Ty_{i,t}}-D_{i,t}\right)^{\left(1-\eta.A_{i,j}\right)}.S_{i,j,k}\right].F_k{}^{l-k}} \tag{8}$$

$$F_k = 1-\omega.T_{i,j,t-1}\ and\ F_1 = \theta \tag{9}$$

The *trust value* for each threat is calculated against the anticipated threat type. $Ty_{k,\mathrm{m}}$stated in equation (8) is the speciality type of the threat. By using the weighted average, a peer is most relied on when the type of threat is in their speciality area. η is coefficient for the hardware based attestation, 0 doesn't consider, 1 relies totally on it. $T_{i,j}$ is the total trust value on *j* observed by *i*. $F_k$is the forgetting factor defined in (9). The *forgetting factor* used in this work is similar to Sun et al.[21]. This makes old experiences fade away and decreases their influence on trust calculation. The more we trust an IDS the faster we forget and the less we trust, the longer we would remember. ω is a coefficient that controls the extent of trust inflence. F_0 is the initial *forgetting factor* and $T_{i,j,0}$ is the initial trust value.

4.3.5. Evaluation

We have created a simulation framework and evaluated our proposed model trying to randomize parameters as much as possible to create a very similar situation as it is in the real world. In all simulations the beta distribution has been used to simulate the decision made to consultation on every peer. The beta function is defined in (11).

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du} \tag{11}$$

$$\alpha = Th_{i,t}\frac{Th_{i,t}.(1-Th_{i,t})}{\gamma.(1-sp_{i,j,n}).D_{i,t}} \tag{12}$$

$$\beta = (1-Th_{i,t}).\frac{Th_{i,t}.(1-Th_{i,t})}{\gamma.(1-sp_{i,j,n}).D_{i,t}} \tag{13}$$

α α and β are set according to the real threat of an incident. Varience of the beta distribution is set to the difficulty of the threat, the speciality of evaluating IDS and a coeffieicnt γ for adjustment. Also the satisfaction equation (5) has been used in all simulations.First we evaluate an IDS with high expertise level to show the fast convergence of the model. In this simulation the attestation of the peer has not been used. The parameters used are listed in table (1).

Table 1 Parameters used in simulation 1

Para	Valu	Description
sp	0.9	The expertise level
$A_{i\ i}$	0	Attestation Value
ρ_1	8	Satisfaction Coefficient
ρ_2	7	Satisfaction difficulty
θ	0.97	Forgetting factor
ω	0.1	Trust influence in
n	1	Number of specialties
γ	0.1	Beta function adjustment
$T_{i\ i\ 0}$	0.5	Initial trust value
F_0	0.5	Initial trust influence
$D_{i\ t}$	0.2-	Difficulty
$Th_{i\ t}$	0.2-	Real threat
$Th_{i\ t}$	Beta	Reported threat

Figure (2) shows the result of first experiment. It shows the first 20 experiences. The reporting error and trust value calculated at that moment has been shown in this figure. We set the trustworthiness threshold to the value of 0.8 that is achieved in the fist 3 transactions.

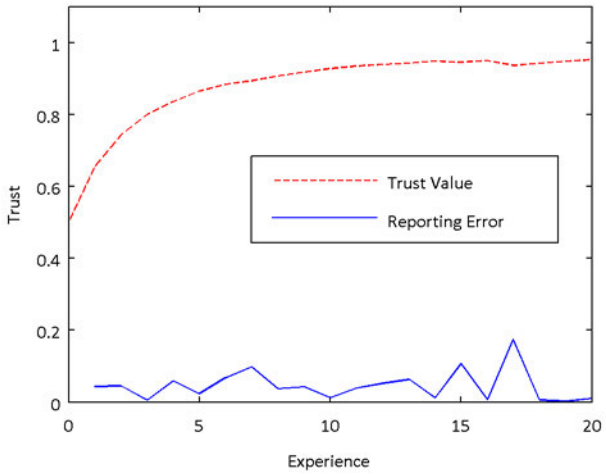

Figure 2 Reporting error and trust value in First Experiment

The final simulation evaluates the trusted computing-based system and its performance against non-trusted computing enabled systems. This time we have two different specializations where the peer is an expert in one and a novice in the other. For each experiment the speciality type of that experiment has been randomly selected between the two. Table (2) show the parameters updated in this simulation from table (1). The simulation results are shown in figure (2).

Table 2 The simulation parameters for second experiment

Para	Valu	Description
sv	0.1.0	The expertise level
$A_{i\ i}$	0,0.9	Attestation Value
n	2	Number of specialties
γ	0.25	Beta function adjustment

The amount of error for consultations where there is little knowledge is known can be significant, but using the weighted average it can be seen that little affect has been put on the trust value. The trusted computing enabled system reaches the trusted value with minimum interactions and is more stable against mistakes made in the evaluation of threats.

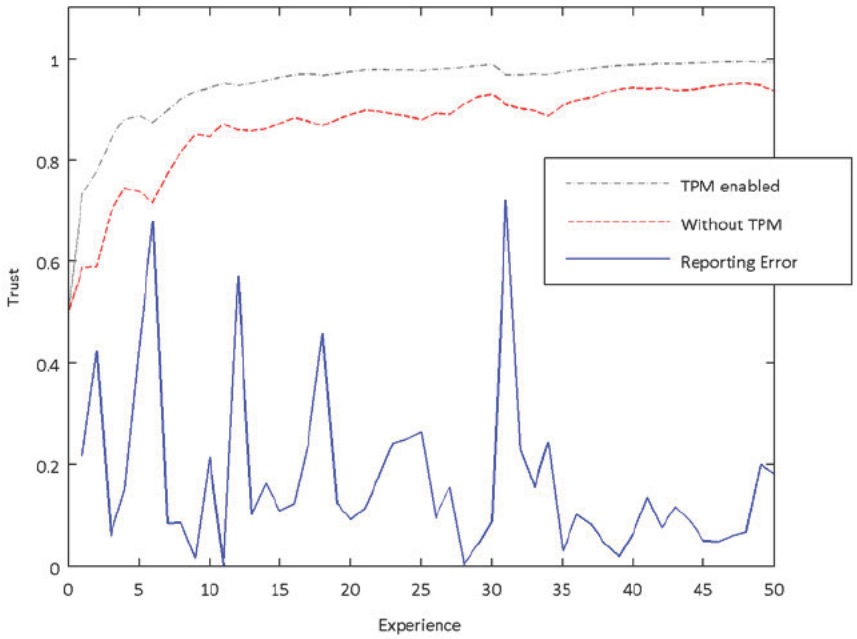

Figure 3 Evaluation of trusted computing based system

5. Conclusion and future work

We have presented in this paper a trust aware Consultative Trusted Computing-based Collaborative IDS (CTC IDS) model that is capable of categorizing each participating IDS expertise (i.e. speciality and competence), which can be used by collaborating organizations to determine the right candidate (IDS node) to deal with any type of attack. We utilize the TPM for integrity evaluation and fine tuning peer IDS evaluation. The proposed CTC IDS model shows that there is reasonable response to a legitimate host that is capable of understanding the expertise needs of any IDS peer. There are many other factors that we have not considered in this work. One of the most influential factors in most attacks is the proximity or closeness of peers. Targeted attacks can easily be detected using this scheme. We also plan to evaluate the model against the proposed threats and evaluate its improved performance. We believe strongly that by using our proposed CIDS Model, most of the defined attacks can be avoided and we are pursuing this research further.

6. Acknoledgement

This research is a collaborative program between University Technology Malaysia and Advanced Analysis and Modeling Cluster, MIMOS Berhad. The financial support by MIMOS Berhad for this work is gratefully acknowledged.

7. References

[1] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," *Fort Washington, PA,* 1980.

[2] http://cert.surfnet.nl/s/2001/S-01-72.htm. (1/3/2009). *SURFnet-CERT S-01-72: Continued Threat of the "Code Red" Worm*.

[3] C. V. Zhou, S. Karunasekera, and C. Leckie, "A peer-to-peer collaborative intrusion detection system," Kuala Lumpur, Malaysia, 2005, pp. 118-123.

[4] C. V. Zhou, S. Karunasekera, and C. Leckie, "Evaluation of a decentralized architecture for large scale collaborative intrusion detection," Munich, Germany, 2007, pp. 80-89.

[5] A. K. Tummala, "Host Bassed intrusion detector: component of distributed

intrusion detection system(DIDS) architecture," PhD, University of texas, 2007.

[6] A. S. Sodiya, H. O. D. Longe, and A. T. Akinwale, "Maintaining privacy in anomaly-based intrusion detection systems," *Information Management and Computer Security,* vol. 13, pp. 72-80, 2005.

[7] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and P2P intrusion detection," West Point, NY, United States, 2005, pp. 333-339.

[8] Naji Habra, B. L. Charlier, A. Mounji, and I. Mathieu, "ASAX:Software Architecture and Rule-Based Language for Universal Audit Trail Analysis," presented at the Proceedings of ESORICS'92, European Symposium on Resarch in Computer Security, November 23-25 Toulouse , Springer-Verlag 1992., 1992.

[9] A. Abraham, R. Jain, J. Thomas, and S. Y. Han, "D-SCIDS: Distributed soft computing intrusion detection system," *Journal of Network and Computer Applications,* vol. 30, pp. 81-98, 2007.

[10] P. G. N. Phillip A. Porras, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," presented at the Computer Science Laboratory,SRI International, 1997.

[11] R. C. Steven Cheung, Mark Dilger, Jeremy Frank, Jim Hoagland, Karl Levitt, JeRowe, Stuart Staniford-Chen, Raymond Yip, Dan Zerkle, "GrIDS (A Graph-Based Intrusion Detection System)," *University of California at Davis,* 1999.

[12] G. S. Mark Crosbie, "AAFID, Active Defense of a Computer System using Autonomous Agents," presented at the Dept of Computer Sciences,Purdue University, 1995.

[13] P. Ning, X. S. Wang, and S. Jajodia, "Modeling requests among cooperating intrusion detection systems," *Computer Communications,* vol. 23, pp. 1702-1715, 2000.

[14] C. J. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust Management for Host-Based Collaborative Intrusion Detection," presented at the Proceedings of the 19th IFIP/IEEE international workshop on Distributed Systems: Operations and Management: Managing Large-Scale Service Deployment, Samos Island, Greece, 2008.

[15] C. J. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *11th IFIP/IEEE International Symposium on Integrated Network Management (IM09)*, ed, 2009.

[16] C. J. Fung, J. Zhang, I. Aib, R. Boutaba, and R. Cohen, "Design of a Simulation Framework to Evaluate Trust Models for Collaborative Intrusion Detection," in *IFIP Network and Service Security Conference (N2S 09)*, ed, 2009.

[17] C. J. Fung, Q. Zhu, R. Boutaba, and T. Barsar, "Bayesian Decision Aggregation in Collaborative Intrusion Detection Networks," in *12th IEEE/IFIP Network Operations and Management Symposium (NOMS10)*, ed, 2010.

[18] L. Chen, R. Landfermann, H. Lohr, M. Rohe, A.-R. Sadeghi, and C. Stuble, "A protocol for property-based attestation," presented at the Proceedings of the first ACM workshop on Scalable trusted computing, Alexandria, Virginia, USA, 2006.

[19] R. Sailer, X. Zhang, T. Jaeger, and L. v. Doorn, "Design and implementation of a TCG-based integrity measurement architecture," presented at the Proceedings of the 13th conference on USENIX Security Symposium - Volume 13, San Diego, CA, 2004.

[20] H. Khorasanizadeh, N. Bashah Idris, and J.-L. Ab Manan, "Trust management in Distributed Intrusion Detection utilizing Platform Integrity measurement," presented at the The International Conference on Cyber Security, CyberWarfare and Digital Forensic, Faculty of Computer Science and Information Technology, University Putra Malaysia, Malaysia 2012.

[21] Y. Sun, Z. Han, and L. K. J. R., "Defense of trust management vulnerabilities in distributed networks," *Communications Magazine, IEEE,* vol. 46, pp. 112-119, 2008.